JOURNAL OF SOFT COMPUTING AND DATA MINING
e-ISSN: 2716-621X

JSCDM

Vol. 6 No. 3 (2025) 33-45
https://publisher.uthm.edu.my/ojs/index.php/jscdm

An Intelligent Botnet Detection System for IoT Using Neural Networks and an Enhanced Moth Search Optimize

Sanaa A. A. Ghaleb[1,2]*, Mumtazimah Mohamad[2,3]*, Waheed A. H. M. Ghanem[1,4,5]*, Abdullah B Nasser[6]

[1] *Faculty of Education, University of Lahej, Lahej, YEMEN*
[2] *Faculty of Informatics and Computing (FIK), Universiti Sultan Zainal Abidin, Kuala Terengganu, 22200 Terengganu, MALAYSIA*
[3] *Artificial Intelligence for Sustainability and Islamic Research Center (AISIR) Universiti Sultan Zainal Abidin, Kuala Terengganu, 22200 Terengganu, MALAYSIA*
[4] *Faculty of Engineering, University of Aden, Aden, YEMEN*
[5] *Faculty of Computer Science and Mathematics (FSKM), Universiti Malaysia Terengganu, Kuala Terengganu, 21030 Terengganu, MALAYSIA*
[6] *School of Technology and Innovation, University of Vaasa, 65200 Vaasa, FINLAND*

*Corresponding Author: sanaaghaleb.sg@gmail.com, mumtza@unisza.edu.my, waheedghanem@umt.edu.my
DOI: https://doi.org/10.30880/jscdm.2025.06.03.003

Article Info

Received: 22 March 2025
Accepted: 14 April 2025
Available online: 30 December 2025

Keywords

Botnet Detection System (BDS), cybersecurity threats, Enhanced Moth Search Algorithm (EMSA), Multi-Layer Perceptron (MLP), classification, Internet of Things (IoT)

Abstract

Botnets, distributed networks of compromised devices under remote control, continue to pose a serious cybersecurity threat, as they are challenging to detect with traditional methods because of their evasion capabilities. The evolving nature of botnets demands stronger detection systems that can effectively detect malicious traffic patterns. To confront this problem, we introduce a new botnet detection framework called BDS (Botnet Detection System), aimed at improving the accuracy of detection and reducing the number of false positive results. We integrate the Enhanced Moth Search Algorithm (EMSA) with Multi-Layer Perceptron (MLP) for the enhancement of training of the neural network model, titled EMSA-MLP. For this, we test our model with a representative Bot-IoT dataset with diverse botnet attack scenarios in terms of separating honest and malicious traffic with the help of efficient optimization by EMSA. To assess the proposed EMSA-MLP, the Bot-IoT benchmark dataset was used, containing a variety of botnet attack methods. The detection performance of the model was demonstrated to be excellent. The model achieved a high detection performance in identifying botnet attacks, with an accuracy rate of 97.09%. They also preserved a good accuracy of 91.59%, and their false alarm rate was kept low at 0.0291. Overall, compared to some common classifiers: random forest, decision tree, and base MLP-this model did pretty well. It also outperformed relatively newer and more complex models. This architecture has achieved good accuracy, while it has not made the growth of IoT settings very large, and it has been formulated to be used by devices. This model provides high accuracy without burdening the IoT infrastructure, so it is applicable to devices with

restricted capabilities. It is suited for practical applications, as it can adapt to different types of data.

1. Introduction

The new challenges of rapidly changing cyber threats and increasing internet complexity provide new openings [1], [2]. Attackers passionately look for weaknesses within the devices, data, applications, and networks and attempt to access the systems at all times. The most serious of the cyber threats are botnets, high-tech chains of hijacked devices managed by a central operator [3], [4]. The structure of a botnet generally comprises three main elements: the botmaster/controller, the compromised devices or bots, and the command-and-control (C&C) framework. Utilizing this structure, the botmaster coordinates large-scale cyber assaults by sending hidden commands, allowing the compromised devices to perform harmful actions without detection [5], [6].

In the rapidly changing landscape of cyber threats, botnets pose an increasing risk to the fundamental principles of cybersecurity, namely confidentiality, integrity, and availability. The fact that they are growing in popularity among hackers is undoubtedly concerning. Botnets are being used in extensive cyberattacks that impair system performance and interfere with essential services, in addition to data breaches. Botnet detection, which is prevalent at both the host and network levels, is essential to network resilience [7]. As pertinent indicators of the machines' pseudo seizures, the host-based approach looks for abnormalities in home resource usage, such as excessive CPU and unusual memory utilization. However, this approach necessitates continuous host monitoring, which is costly and challenging to scale.

On the other hand, analyzing communication patterns and data movement towards the network is the role of targeted technologies, by identifying the activity of botnets [8]. This approach, in contrast to host-oriented approaches, remains effective even when attackers employ encryption to hide their actions. The signature-based approach utilizes deep packet inspection (DPI) to identify specific botnet patterns, leading to high detection accuracy and minimal false positives [9]. Nevertheless, this method is inadequate against new threats, as it necessitates regular updates to recognize new botnet variations. As botnets evolve to be more adaptable, conventional detection techniques face challenges, highlighting the necessity for innovative and dynamic strategies to address this growing cyber threat [10].

Moreover, encryption techniques introduce additional layers of complexity to detection by obscuring harmful signatures, which diminishes the effectiveness of conventional identification methods. Consequently, techniques that concentrate on identifying anomalies in botnet activity and packet payload sizes have surfaced. However, because botnets are constantly changing and adapting their attack strategies, it is becoming more difficult to identify them precisely [11]. To tackle this problem, metaheuristic techniques have gained popularity due to their ability to uncover hidden anomalies in network traffic. Anomaly detection systems (ADS) are prone to high FPRs and classifications, while traditional ML techniques are limited by their computational resources, long training times, and extensive feature engineering. Current ML and metaheuristic optimization techniques still face numerous challenges, despite the significant advancements in botnet detection.

Prior works have explored techniques such as NB classifiers [12], SVMs [13], and ANNs [14], and all of them suffered from either cost or performance. Hybrid methods are also being investigated to enhance the accuracy of MLPs, as the analysis indicates that MLPs exhibit favorable performance in detecting complex attack patterns. Conventional training methods, such as backpropagation, are susceptible to local minima, have a lengthy convergence time, and demand flexibility. Other techniques for identifying botnets still require improvement [15]. We urgently need more adaptable, effective, and scalable optimization techniques in light of these enduring issues. As these methods require less processing power, they should facilitate botnet detection (BD).

We present EMSA-MLP, a novel machine learning technique that enhances MLP classifier training by utilizing the EMSA [16]. EMSA-MLP is a dynamic optimization technique that addresses key issues like slow convergence, becoming trapped in local minima, and using excessive processing resources while speeding up learning. The gradient-free optimization process, adaptive search strategies that successfully balance exploration and exploitation, and the small number of control parameters of our approach make it reliable and scalable for real-world botnet detection applications. We use the Bot-IoT Benchmark dataset, a large botnet attack dataset, to evaluate the effectiveness of EMSA-MLP.

Our main goal is to enhance the accuracy of BD while minimizing false positives, thus fortifying cybersecurity measures. This research enhances MLP performance using advanced EMSA algorithms for BD and evaluates its effectiveness using the Bot-IoT dataset for emerging threat identification. The following sections will cover the subsequent topics: Section 2 will review related work. Section 3 will provide an overview of the proposed BDS and MSA and discuss the adaptation of EMSA for MLP training. Section 4 will showcase the performance evaluation and ensuing discussion. Section 5 will analyze the experiments conducted. Lastly, Section 6 will wrap up the investigation.

J. of Soft Computing and Data Mining Vol. 6 No. 3 (2025) p. 33-45

2. Literature Review

This section offers a thorough overview of important studies that have concentrated on training MLPs using probabilistic methods. The basis for improving the MSA, which is now the foundation for optimizing MLP training in our current BD research, was established by our previous work [16]. Researchers have increasingly used stochastic global optimization (SGO) techniques in recent years to enhance MLP training. These techniques facilitate the solution of difficult optimization problems by generating a large number of random solutions. Nature-inspired meta-inference algorithms (NIMA) have gained widespread popularity among various SGO strategies due to their ability to increase learning efficiency by mimicking biological and natural processes, making them an effective tool for training neural networks [17]. Swarm intelligence (SI) is a particularly intriguing subset of NIMAs that provides a dynamic substitute for conventional trajectory-driven NN training. The extremely complicated and nonlinear error landscapes of NNs, where multiple local minima can trap the learning process, are frequently too difficult for traditional trajectory-based approaches, which concentrate on minimizing error functions. Therefore, rather than striving for complete global optimization, these methods often settle on suboptimal solutions. The true benefit of SI techniques lies in their adaptability and flexibility. These methods can significantly speed up the training process, dynamically add new data to improve learning, and effectively handle repetitive training patterns, setting them apart from traditional methods. Because of these characteristics, they can successfully address the changing issues in NNs optimization, which helps to produce BDs that are more accurate and efficient.

In [18], the study introduced an optimized hybrid classification framework for IoT botnet detection, which combined RNN and Improved Deep Belief Network (IDBN) with Self-Adaptive Beluga Whale Optimization (SA-BWO) to accurately identify attacks. Feature extraction techniques like statistical measures and correlation-based insights were employed to increase detection accuracy, while batch normalization and dropout layers prevented overfitting. Experimental validation on IoT botnet datasets demonstrated superior performance in comparison to conventional detection methods.

In [19], the study of deep context near field communication (NFC), a deep learning framework, was released to enhance NFC security in IoT ecosystems against threats such as spoofing and eavesdropping. The framework consists of an adaptive threat feedback system for real-time threat detection, a privacy masking layer for privacy enforcement, and a context encoder for pattern recognition.

CNN-DSA, a deep learning-based botnet detection model for IoT, was introduced in [20] and combines Convolutional Neural Network (CNN) and deep stacked autoencoder (DSA). The model select feature using city block distance and information gain, preprocesses log data using quantile normalization, and enhances the data using oversampling.

To tackle changing attack patterns, in [15] this research introduces a neural for IoT networks that is based on federated learning. The Chimp optimization technique and deep learning classifiers are used to increase detection accuracy. A new BD model uses the EMSA to improve detection accuracy for botnets in IoT. It shows strong performance in tests with the benchmark dataset. A summary table of related research is included. Table 7 presents a comprehensive list of all commonly used notations.

Table 1 *The summary overview of the related work in 2025*

Ref.	Methodology	Results/ Accuracy	Demerits
[18]	SA-BWO for RNN	94.0	Despite its high accuracy, the proposed framework has limitations, such as increased computational complexity, potential overfitting, and reliance on extensive training data. Additionally, SA-BWO optimization may introduce processing overhead, hindering real-time implementation in resource-constrained IoT environments.
[19]	NFC deep learning framework for NFC security in IoT	94	NFC faces high computational costs, risk of overfitting, and scalability challenges in large IoT networks.
[20]	Deep learning-based botnet detection using CNN-DSA	92.4	The CNN-DSA model is computationally expensive, struggles with imbalanced data, and is unsuitable for low-power IoT devices.

Ref.	Methodology	Results/ Accuracy	Demerits
[15]	Deep learning and Chimp optimization for feature selection	CNN 95, DT 91, MLP 90, NB 64, RF 91	The approach faces challenges such as high communication overhead, vulnerability to poisoning attacks, and dependence on high-quality distributed data.

3. Methodology

This study aims to devise and implement an ANN that has been trained using the advanced Enhanced Moth Search Algorithm (EMSA), as depicted in Figure 1. We aim to develop a novel framework that demonstrates outstanding performance in terms of accurately classifying, efficiently detecting, minimizing false alarms, achieving global convergence, and maintaining robustness in detecting attacks. This is achieved through the utilization of EMSA during the training of ANNs. Multilayer perceptron neural networks (MLPNN) are robust classification tools renowned for their effectiveness in addressing the complexities encountered in botnet detection systems. They excel at identifying the standard attributes of system users within the Internet of Things (IoT) context and detecting noteworthy deviations from established user behaviors with statistical significance. These networks can build a thorough knowledge model of environmental behavior because of their adaptable and scalable structures. The first stage is setting up the neural network's architecture and variables to make it easier to learn how input patterns relate to the intended outcome during training. In domains where knowledge acquisition is essential, training neural networks such as MLPs is a crucial and challenging task. In a nonlinear optimization framework, the training process can be thought of as an optimization effort in which the solution is obtained within the parameters of linear constraints. As a result, different optimization strategies have been investigated in the literature to successfully address this problem.

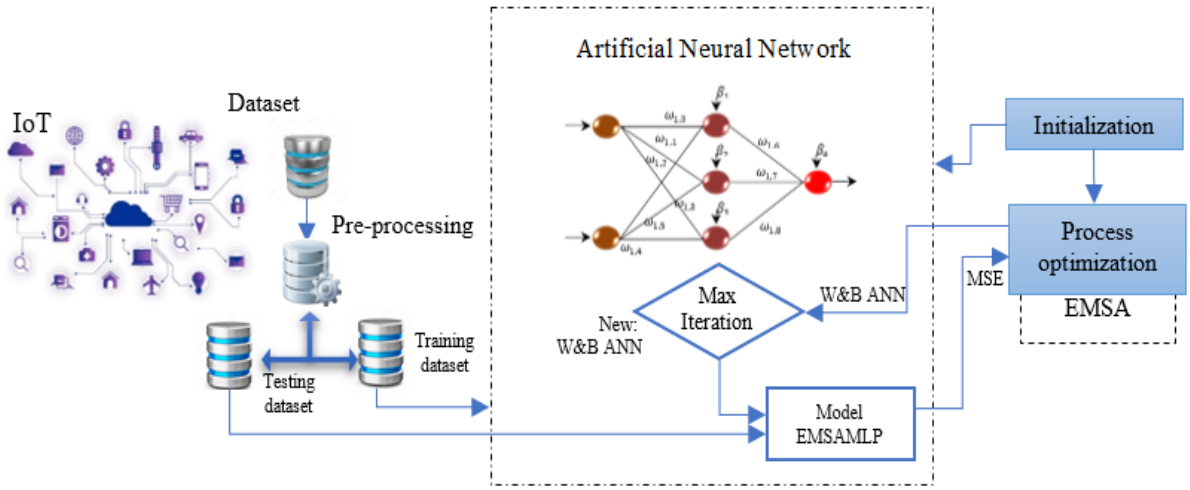

Fig.1 *The EMSAMLP-BDS model*

The improved MSA method is a stochastic optimization strategy aimed at discovering viable or near-optimal solutions within a specified exploration area. EMSA addresses the challenge of being trapped in local best solutions within the exploration area, guaranteeing swift progress towards the overall best solution while upholding resilience and universal convergence capabilities. In Figure 1, we illustrate our botnet detection system model, encompassing several components: the dataset module for botnets, the NN module, and the module for optimization. These modules are detailed as follows: The initial phase of the suggested model involves utilizing the botnet dataset component (BD) to manage, filter, and extract attributes from the audit information. This dataset consists of predefined training and testing collections; they serve as inputs for the ANN component that follows. Prior to supplying the neural network component with data, it's crucial to standardize the input data within the range of [-1, +1] to prepare it for the following component.

The following step involves employing the ANN part, which takes in the extracted features from the botnet dataset. This ANN part is structured as an MLPNN, comprising an input layer, multiple hidden layers with specified neuron quantities, and an output layer. The data from the botnet dataset (referred to as the training dataset) is inputted into the ANN to serve as a training model for improving the network. During this training process, mean squared error (MSE) is utilized as the metric for evaluating errors, a metric used by the optimization component (EMSA module) to adjust the architecture, w, and B of the network. The EMSA module functions autonomously to

modify the structural setup and the parameters (weights/biases) following each iteration. During each training iteration, the EMSA module transmits its setups, comprising structural intricacies and parameter sets, to the ANN module. Subsequently, the ANN module evaluates these setups using a training dataset and delivers their respective performance evaluations. In this study, the MSE served as the performance metric for the novel EMSA training technique. The architecture and parameters of the ANN are determined through minimizing the MSE. This leads to an update of the knowledge base, which encompasses the ANN's configuration and parameters. The "Max Iteration" parameter illustrated in Figure 1 governs when the training halts. The final stage involves choosing the most effective model with the best ANN architecture and parameters, established through the training data. In this stage, the test inputs are sourced from a separate dataset and fed into the trained ANN for prediction. The testing phase of the ANN involves comparing the predicted outputs with the closest matches among the target categories.

3.1 Moth Search Algorithm (MSA)

3.1.1 Lévy Flights

To navigate toward the optimal butterfly, the moth employs Lévy flights, leveraging its proximity to the target. This movement results in position updates, as defined in Eq. (1). The Lévy distribution governing this process can be mathematically expressed using a power-law formula, as outlined by [21] and illustrated in Eq. (2).

$$x_i^{t+1} = x_i^t + \alpha L(s) \tag{1}$$

$$L(s) \sim |s|^{-B} \quad \text{where } 1 < \beta \leq 3 \text{ is an index} \tag{2}$$

In mathematical or computational modeling, particularly in optimization or evolutionary algorithms, a parameterized location update method is utilized during the generation-based process. This method involves updating positions iteratively across generations. For instance, variables like t denote the current generation, x_i^{t+1} and x_i^t represent the updated and previous positions of i^{th} individual at generation t, respectively. $L(s)$ for the step size determined from Lévy flights are all included in the notation. Furthermore, the parameter α, whose precise description is given within the context of the study, represents the scale factor for the problem.

$$\alpha = S_{max}/t^2 \tag{3}$$

The value of S_{max}, which stands for the largest step size for movement in this context, is established by the specifications of the task at hand. Equation (1) formulates the Lévy distribution, denoted as $L(s)$, as follows:

$$L(s)=\frac{(\beta-1)\Gamma(\beta-1)sin(\frac{\pi(\beta-1)}{2})}{\pi S^{\beta}} \tag{4}$$

When the gamma function, $L(x)$, exceeds s, the Lévy distribution with a parameter value of β = 1.5 serves as a basis for determining the Lévy flights performed by moths, as previously explained.

3.1.2 Fly Straightly

When moths are at a considerable distance from a light source, they may travel in a straight path toward it. This movement pattern can be characterized as follows:

$$x_i^{t+1} = \lambda \times (x_i^t + \phi \times (x_{best}^t - x_i^t)) \tag{5}$$

In this method, ϕ serves as a scaling factor derived from the golden ratio, while x_{best}^t represents the leading moth at generation t. The parameter λ acts as a scale factor. For simplicity, either Equations (5) or (6) will be applied to update the position of moth i with a 50% probability. Alternatively, the moth may move toward its prior position, increasing its distance from the light source. In this case, the final position of moth i can be expressed as follows:

$$x_i^{t+1} = \lambda \times (x_i^t + \frac{1}{\emptyset} \times (x_{best}^t - x_i^t)) \tag{6}$$

Three primary positions: x_{best}, x_i, and $x_{i,\,new}$, representing the optimum, initial, and adjusted positions of a moth navigating toward a light source. x_{best} signifies the most favorable position, x_i indicates the moth's starting point, and $x_{i,\,new}$ reflects an updated location following an optimization process. To make things better, whether that

means getting lighter to attract them, using less energy, or enhancing a specific goal metric, the transfer from x_i to $x_{i,\ new}$ new requires an optimization process. The moth moves around in a loop, getting closer and closer to x_{best}. The moth starts at x_i, checks on itself, goes closer to x_{best}, and then starts the cycle again in what is called a feedback loop. This iterative improvement is what makes it possible to keep learning and adapt. A good optimization and goal achievement should be indicated by the moth ideally convergent towards x_{best} after a few modifications. The parameter λ is an essential scaling factor that directly affects the rate of convergence of the algorithm and increases population diversity. From a conventional uniform distribution, a randomly chosen value is assigned to this scaling factor. Algorithm also offers the EMSA pseudocode, which outlines its organizational structure and implementation procedures.

EMSA Algorithm

Step 1: Initialization
Put t = 1 in the generation counter
Using a uniform distribution, start a population P of NP moths at random
Define key parameters: MaxGen, Smax, β, φ
Create a random initial population (X_i^d) at random using the following formula:
For each moth i from 1 to N
For each dimension d
Assign a random value to X_i^d
Step 2: Fitness Evaluation
Determine each moth's fitness value
Identify the best-performing moth $\widehat{T_d}$
set $\widehat{T_d}$ as the initial best solution
Step 3: Optimization Process
While t < MaxGen, do the following:
Update the solution based on Eqs. (7) and (8)
For each moth i in the population:
If ε1 ≤ p // Execute the MSA phase
Else if ε2 ≤ limit
Randomly select two moths r1 and r2
Update the position of x_i^{t+1} as:
$x_{ir_1}^{t+1} = x_{r_1}^t$
$x_{ir_2}^{t+1} = x_{r_2}^t$
Based on selected moths, update position:
If r1 ≠ r2
Use best solution update:
$x_i^{t+1} = w \times (x_{ir_1}^t - x_{best}^t) \times 2 \times (rand-1);$
Otherwise:
Use alternative update:
$x_i^{t+1} = w \times (x_{ir_2}^t - x_{best}^t) \times 2 \times (rand-1);$
If neither condition is met:
Use:
$x_i^{t+1} = x_{min}^t + rand \times (x_{max}^t - x_{min}^t);$
Update the best solution T
Increment t by 1
Step 4: Termination
Return the best solution T

To improve its performance, the EMSA algorithm incorporates a mutation operator, setting it apart from the MSA. This improvement increases population diversity by introducing novel solutions in every iteration. The mutation operator is essential for increasing search effectiveness and accelerating convergence to the best answer.

EMSA adds two more control variables, *p* and *limit*, while maintaining all of the MSA's characteristics. With *p* being a value between 0 and 1 and the *limit* being established by two random numbers selected from a uniform distribution, these parameters are essential for preserving equilibrium. The MSA phase is used to produce a new solution if $\varepsilon_1 \leq p$. On the other hand, the mutation operator is used to produce a new solution if $\varepsilon_1 > p$. The mutation operator is controlled by the *limit* parameter. In the interval [1, N], r_1 and r_2 are distinct values, and *N* is the population size. If $\varepsilon_2 \leq Limit$ two moths, $x_{r_1}^t$ *and* $x_{r_2}^t$ are selected at random from the population.

J. of Soft Computing and Data Mining Vol. 6 No. 3 (2025) p. 33-45

Equation (7) updates x_i^{t+1} when $r_1 \neq r_2$, while Equation (8) ensures no modifications if they are the same. If $\varepsilon_2 >$ *limit*, $x_i^{t+1}$is randomly chosen from the set of feasible solutions. Within the population, the best performing moth is denoted as x_{best}^t, where t refers to the current generation. Random numbers with a uniform distribution between 0 and 1 are produced by a variable rand, further enhancing the algorithm's stochastic behavior.

$$x_i^{t+1} = w \times \left(x_{ir_2}^t - x_{best}^t\right) \times 2 \times (rand - 1); \tag{7}$$

$$x_i^{t+1} = w \times \left(x_{ir_1}^t - x_{best}^t\right) \times 2 \times (rand - 1); \tag{8}$$

EMSA enhances the capabilities of MSA by integrating an innovative mutation operator. This addition increases population diversity, promoting a wider spectrum of genetic variations. The primary goal is to improve MSA's efficiency, accelerating convergence toward the optimal solution.

3.2 The EMSA Adaptation Process

In optimization methodologies, the fitness function acts as the objective function, with the goal being to minimize the values it produces. This objective aligns with research on training techniques [22], [23].
As illustrated in Figure 2, while working with *N* input nodes, *O* output nodes and, the output for the i^{th} hidden node is calculated as follows:

$$1 \Big/ \left(1 + exp\left(-\left(\sum_{i=1}^{N} \mathcal{W}_{ij}.\mathcal{X}_i - \beta_j\right)\right)\right), j = 1, 2, \dots, H \tag{9}$$

$S_j = \sum_{i=1}^{N} \mathcal{W}_{ij}.\mathcal{X}_i - \beta_j, S_j$ indicates the total of the products of $\mathcal{X}_i$ which stands for the i^{th} input and $\mathcal{W}_{ij}$ which indicates the weight assigned to the i^{th} input node to the j^{th} hidden node. Furthermore β_j represents the bias of the j^{th} hidden node. The outcome in the end can be stated as follows:

$$\mathcal{O}_k = \sum_{i=1}^{N} \mathcal{W}_{kj}.f\left(S_j\right) - \beta_k, k = 1, 2, \dots, O, \tag{10}$$

β_k is the bias (threshold) of the k^{th} output node, indicated as k^{th}, and $\mathcal{W}_{kj}$ is the weight associated with the j^{th} hidden node that leads to the k^{th} output node. The learning error, which serves as the fitness function, can be expressed as follows:

$$E_k = \sum_{i=1}^{O} \left(\mathcal{O}_i^k - d_i^k\right)^2 \tag{11}$$

$$MSE = \sum_{k=1}^{q} \frac{E_k}{q} \tag{12}$$

Where q represents the training samples, $\mathcal{O}_i^k$ represents the actual output of the i^{th} input during the utilization of the k^{th} training sample, and d_i^k represents the expected output of the i^{th} input unit for the k^{th} training sample. As a result, the fitness function for the i^{th} training sample can be represented as follows:

$$Fitness\ (x_i) = MSE\ (x_i) \tag{13}$$

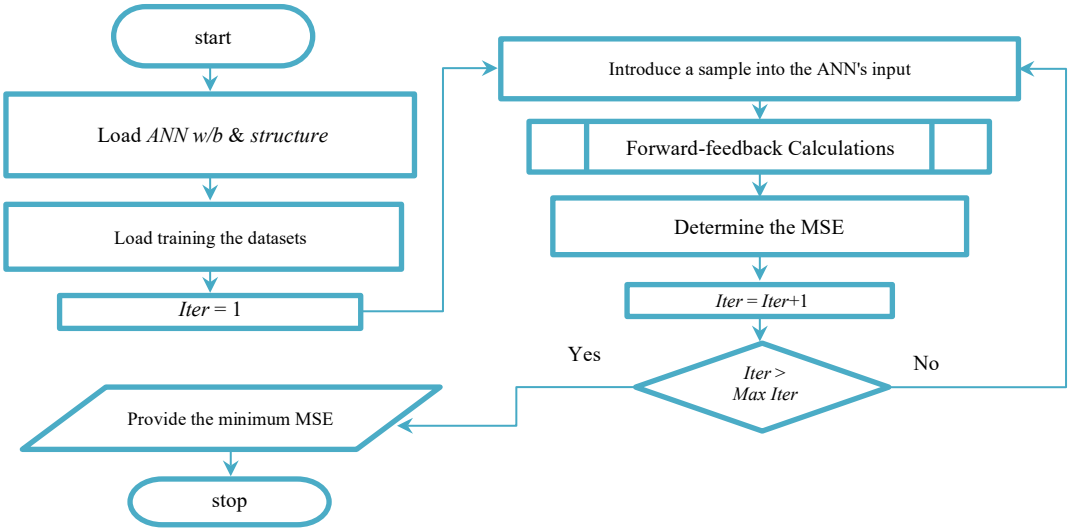

Fig. 2 *Flowchart for ANN model training*

4. Performance Evaluation and Discussion

All comparative experiments involving model evaluations were carried out on a PC equipped with an Intel Core i5 processor, running MATLAB R2014a on the Microsoft Windows 8 operating system. These studies were conducted without the use of proprietary software. The model's performance is evaluated using a dataset that focusses on botnet activity and separates attack and non-attack cases. It was created at UNSW Canberra's Cyber Range Lab, simulating a real-world internet connection that includes malicious and lawful data [24], [25].

Table 2 *All features of the dataset*

Name	Description	Name	Description
Pk Seq ID	Unique Row ID	N N Conn PSrc IP	Number of incoming connections for every IP source
Pro to	Network Protocol	Min	Shortest duration among aggregated records
Sad dr	Sender IP Address	State number	Numeric encoding of feature status
S port	Sender Port Number	Mean	Mean duration of aggregated records
Dad dr	Receiver IP Address	NINConnP Dst IP	Number of connections coming in for each destination ID
D port	Receiver Port Number	D rate	The number of packets per second that are sent from the source to the destination
seq	Packet Sequence Number	S rate	Sending rate (packets/sec) of packets from source to destination
Std dev	Standard Deviation of Collected Records	max	Longest duration among aggregated records
D rate	Packets per Second	NIN Conn PSrc IP	The number of incoming connections linked to a source IP
S rate	Packets per Second	min	Minimum amount of time noticed in total records
max	Longest Duration Among Aggregated Records	State number	A numerical depiction of the feature condition
NINConn P Dst IP	Count of Incoming Connections per Destination ID	mean	Average length of time observed across all records

The dataset was arbitrarily divided into 70% training and 30% testing data. Comprising multiple formats such as PCAP, Argus, and CSV, the dataset is meticulously categorized by attack type and subcategory to enhance classification accuracy. The CSV files contain 16.7 GB of extracted flow traffic, complementing 69.3 GB of PCAP data, which holds over 72 million records. To simplified dataset management, MySQL queries were applied to filter out 5% of the original data, resulting in a refined dataset comprising around 3 million records distributed

J. of Soft Computing and Data Mining Vol. 6 No. 3 (2025) p. 33-45

across four files, with a total size of 1.07 GB. In total, the dataset consists of 3.2 million instances, with only 477 representing legitimate traffic. Each instance is assigned a classification label, where 0 signifies non-attack traffic, while 1 indicates an attack. A comprehensive overview of the dataset's features is provided in Table 2. Several algorithms have been analyzed to assess the reliability of the new model. Identical values were assigned to all algorithm control variables, including the solution and the dimensionality of the BDS, which represents the dataset's features. The parameters of the algorithms utilized in this training are detailed in Table 3.

Table 3 *Lists the parameters for the algorithms used in this investigation*

Alg.	Descriptions	Value	Alg.	Descriptions	Value
MSA	the maximum generation	2	DA	-	[0, 1] - 1.5
	max walk step	1.0		separation weight	0.1
	index B	1.5		inertia weight	0.2-0.9
	acceleration factor	0.618		alignment weight	0.1
ALO	linear decreased	2	MBO	butterfly adjusting rate	0.4167
	random walk	[0, 1]		max step	1.0
DA	cohesion weight	0.7		migration period	1.2
	food factor	1		migration ratio	0.4
	enemy factor	1			

The practicality of the suggested model was assessed and measured using standard criteria, such as accuracy (Acc), commonly employed to assess the efficiency of EMSA-MLP. These criteria include parameters like true negatives (TN), true positives (TP), false negatives (FN), and false positives (FP) rates. Table 4 showcases the corresponding performance indicators.

Table 4 *Matrix of performance for classification*

Standard Metrics	Descriptions	
Accuracy (ACC)	$ACC = ((TP + TN)/(TP + TN + FP + FN))$	(14)
False alarm rate (FAR)	$FAR = (FP/(FN + TN))$	(15)
Detection rate (DR)	$DR = (TP/(TP + FP))$	(16)

5. Results and Discussion

This study evaluates a proposed model by comparing it with standard metrics such as Acc., DR, and FAR, as commonly used in the field. We utilized the enhanced Moth Search Algorithm (EMSA) from our previous study [16] to train Multilayer Perceptron (MLP) models, specifically EMSA-MLP, which combines EMSA techniques with MLP. The main goal is to use MLP for botnet detection. The assessment in this section focuses on this model using standard botnet dataset. We will first describe the general setup employed for all experiments and then delve into our results. Initially, experiments and comparative assessments were conducted on five algorithms, with the proposed algorithm being tested on a botnet dataset. The classification process is significantly influenced by the dataset, so variations in data subsets can yield distinct outcomes, particularly when the dataset includes novel attacks.

Therefore, the proposed approach underwent testing with diverse datasets to ensure a dependable evaluation. These datasets, ranging from older to newer ones, were extensively discussed in Section (4). Each method was utilized to train the multilayer perception and subsequently tested using the aforementioned dataset (different training and testing dataset were used in our investigation.). Additionally, the outcomes of each experiment are illustrated in a chart to visually depict how well each algorithm converges. Moreover, a table is provided that details the numerical outcomes and performance indicators for each model. A maximum of 100 iterations were performed on each model, with results derived from 100 runs.

5.1 Scenario 1: Performance of EMSA-MLP on the Dataset

The effectiveness of the EMSA-MLP model, derived from MLP, was verified through experimental trials in Scenario 1. Table 5 illustrates the comparative outcomes of employing the EMSA-MLP model with the Bot-IoT botnet detection benchmark dataset. All calculations for the proposed EMSA-MLP models were made using the definitions specified in Table 4 and Eqs. (14-16).

Table 5 *The metrics for the five models' performance on the Bot-IoT dataset*

No.	Alg.	Acc.	DR	FAR	RACC	RDR	RFAR
1	EMSAMLP	97.09	91.59	0.0291	1	1	1
2	MSAMLP	96.87	90.65	0.0313	2	2	2
3	ALOMLP	96.74	88.79	0.0326	3	4	3
4	MBOMLP	95.98	90.65	0.0402	4	2	4
5	DAMLP	92.16	81.31	0.0784	5	5	5

Table 5 presents the performance metrics of five models evaluated on the Bot-IoT dataset, with emphasis on Acc., DR, FAR, and ranking metrics (RACC, RDR, RFAR). The EMSAMLP model outperforms the others with the highest Acc. (97.09%) and DR (91.59%), while also maintaining the lowest false alarm rate (0.0291), securing the top rank in all ranking categories. MSAMLP follows closely with a (96.87%) accuracy and (90.65%) detection rate, ranking second overall. ALOMLP achieves (96.74%) accuracy but has a lower detection rate (88.79%) and a slightly higher FAR (0.0326), placing it third. MBOMLP performs slightly worse with (95.98%) accuracy and the highest FAR among the top models (0.0402), ranking fourth. DAMLP has the lowest accuracy (92.16%) and detection rate (81.31%), with the highest false alarm rate (0.0784), making it the least effective model.

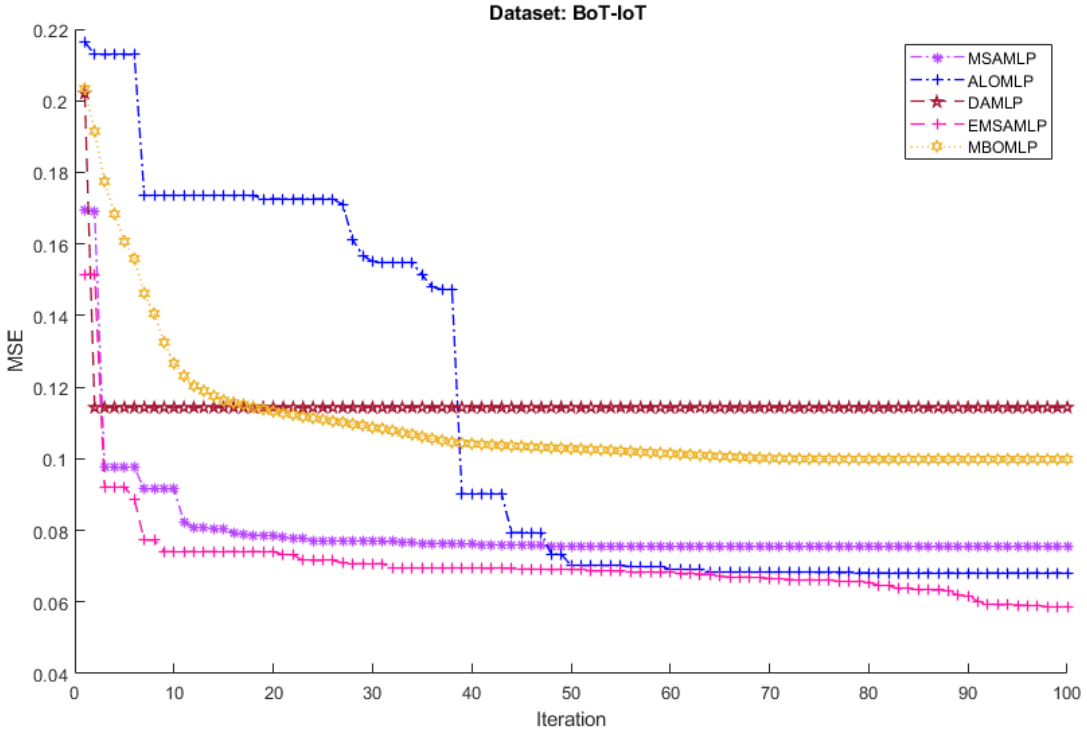

Fig. 3 *The averages of MSE on the dataset*

Figure 3 explains that the graph presents the MSE performance of five machine learning models: EMSAMLP, ALOMLP, MBOMLP, MSAMLP, and DAMLP, over 100 iterations using the dataset. The goal is to minimize MSE, indicating improved predictive accuracy. EMSAMLP (pink) demonstrates the fastest convergence, reaching a low MSE within the first 30 iterations, highlighting its superior learning efficiency. MSAMLP (purple) and ALOMLP (blue) show a gradual decrease in MSE, with MSAMLP experiencing fluctuations before stabilizing after 50 iterations. MBOMLP (yellow) remains nearly constant at a moderate MSE, suggesting limited learning improvement. DAMLP (red) exhibits the highest and most stable MSE, making it the least effective model. Overall, EMSAMLP achieves the fastest convergence, while DAMLP shows the weakest performance.

5.2 Scenario 2: Comparison of Performance Between Proposed Methods and Others

In Scenario 2, we compared our model's performance with newer botnet detection systems listed in Table 6. We used six state-of-the-art methods from [18], [19], [20], and [15] to conduct experiments on the dataset. Table 6 displays the comparison with other existing studies using the same dataset. This method outperforms the others

in terms of accuracy score. Our model outperforms others on Bot-IoT datasets, showcasing its superiority and contribution. It excels in evaluation criteria, accurately classifying data unlike static schemes.

Table 6 *Results comparison with other well-known methods*

References	Method	Metrics	Results	References	Method	Metrics	Results
[18]	SA-BWO RNN	Accuracy	94.0	[20]	CNN-DSA	Accuracy	92.4
[19]	NFC		94	[15]	DT, NB, RF, MLP		91,64, 91,90
This work	EMSA-MLP		97.09	This work	MSA-MLP		96.87

6. Conclusion

The proposed EMSA-MLP model demonstrated superior performance in detecting IoT botnets by leveraging EMSA for optimizing MLP training. Compared to existing models, EMSA-MLP achieved a remarkable 97.09 classification accuracy on the Bot-IoT benchmark dataset, highlighting its effectiveness in botnet detection. While EMSA enhances model training efficiency, its application to a single dataset leaves room for further improvements. Future work could focus on integrating feature reduction techniques to optimize model performance and extend its applicability across diverse botnet detection scenarios, ultimately improving detection accuracy and computational efficiency.

Acknowledgement

This research was supported by the Universiti Malaysia Terengganu (UMT/TAPE RG 2020/55225) & Artificial Intelligence Research Interest Group (AIRIG), also supported by the Center of Research Excellence and Incubation Management (CRIEM) and the Artificial Intelligence Research Centre for Islam and Sustainability (AIRIS) of Universiti Sultan Zainal Abidin, Terengganu, Malaysia.

Conflict of Interest

The authors declare that they have no financial, personal, or professional conflicts of interest that could influence or bias the execution, findings, or interpretation of this research. They confirm the absence of any affiliations or relationships that may be perceived as potential conflicts concerning this study. Any sources of funding, if applicable, have been transparently disclosed, and all contributors and collaborators have been appropriately acknowledged. Furthermore, the authors affirm that this research was conducted with integrity and in compliance with the ethical standards relevant to the field.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design****: Sanaa A. A. Ghaleb, Mumtazimah Mohamad, Waheed A. H. M. Ghanem;* ***data collection****: Sanaa A. A. Ghaleb, Abdullah B Nasser****; analysis and interpretation of results****: Sanaa A. A. Ghaleb, Waheed A. H. M. Ghanem, Abdullah B Nasser. All authors reviewed the results and approved the final version of the manuscript*.

Table 7 *Notations*

Notation	Description	Notation	Description
NIMAs	Nature-Inspired Metaheuristic Algorithms	RNN	Recurrent Neural Network
MSA	Moth Search Algorithm	NFC	Near Field Communication
EMSA	Enhanced Moth Search Algorithm	CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron	DSA	Deep Stacked Autoencoder
BDS	Botnet Detection System	MSE	Mean Squared Error
IoT	Multi-Layer Perceptron	IDBN	Improved Deep Belief Network
C&C	Command-and-Control	SI	Swarm Intelligence
CPU	Central Processing Unit	NN	Neural Network
DPI	Deep Packet Inspection	SGO	Stochastic Global Optimization
ML	Machine Learning	ADS	Anomaly Detection Systems
NB	Naïve Bayes Classifiers	ANN	Artificial Neural Network
SVM	Support Vector Machines	RNN	Recurrent Neural Network
SA-BWO	Self-Adaptive Beluga Whale Optimization	EMSA- MLP	Enhanced Moth Search Algorithm- Multi-Layer Perceptron
RF	Random Forest		
DT	Decision Tree		

References

[1] Bederna, Z., & Szadeczky, T. (2020). Cyber espionage through Botnets. Security Journal, 33(1), 43-62.

[2] Wong, P. Y., Alduais, N. A. M., Mahdin, H., Saad, A. M. H., Abdul-Qawy, A. S. H., Nasser, A. B., & Ghanem, W. A. H. (2024). An Efficient MCD-OSVM Model for Outlier Detection in IoT-Based Smart Energy Management Systems. Journal of Soft Computing and Data Mining, 5(2), 1-14.

[3] Shaaban, M. F., & Ali, A. (2025). An efficient convolutional neural network based attack detection for smart grid in 5G-IOT. International Journal of Critical Infrastructure Protection, 48, 100738.

[4] Nourildean, S. W., Mefteh, W., & Frihida, A. M. (2025). An Artificial Intelligence of Things Intrusion Detection Framework for Mitigating Cyber and Ransomware Threats in IoT Networks. Journal of Soft Computing and Data Mining, 6(1), 333-346.

[5] Nasser, A. B., Ghanem, W. A. H., Saad, A. M. H., Abdul-Qawy, A. S. H., Ghaleb, S. A., Alduais, N. A. M., ... & Ghetas, M. (2024). Depth linear discrimination-oriented feature selection method based on adaptive sine cosine algorithm for software defect prediction. Expert Systems with Applications, 253, 124266.

[6] Abood, M. J. K., & Abdul-Majeed, G. H. (2024). Enhancing multi-class ddos attack classification using machine learning techniques. J Adv Res Appl Sci Eng Technol, 43(2), 75-92.

[7] Padhiar, S., & Patel, R. (2023). Performance evaluation of botnet detection using machine learning techniques. International Journal of Electrical and Computer Engineering (IJECE), 13(6), 6827-6835.

[8] Alqahtani, M., Mathkour, H., & Ben Ismail, M. M. (2020). IoT botnet attack detection based on optimized extreme gradient boosting and feature selection. Sensors, 20(21), 6336.

[9] McCarthy, A., Ghadafi, E., Andriotis, P., & Legg, P. (2022). Functionality-preserving adversarial machine learning for robust classification in cybersecurity and intrusion detection domains: A survey. Journal of Cybersecurity and Privacy, 2(1), 154-190.

[10] Gelgi, M., Guan, Y., Arunachala, S., Samba Siva Rao, M., & Dragoni, N. (2024). Systematic literature review of IoT botnet DDOS attacks and evaluation of detection techniques. Sensors, 24(11), 3571.

[11] Randhawa, R. H., Aslam, N., Alauthman, M., Rafiq, H., & Comeau, F. (2021). Security hardening of botnet detectors using generative adversarial networks. IEEE Access, 9, 78276-78292.

[12] Mehmood, A., Mukherjee, M., Ahmed, S. H., Song, H., & Malik, K. M. (2018). NBC-MAIDS: Naïve Bayesian classification technique in multi-agent system-enriched IDS for securing IoT against DDoS attacks. The Journal of Supercomputing, 74(10), 5156-5170.

[13] Ioannou, C., & Vassiliou, V. (2021). Network attack classification in IoT using support vector machines. Journal of sensor and actuator networks, 10(3), 58.

[14] Gopi, R., Sathiyamoorthi, V., Selvakumar, S., Manikandan, R., Chatterjee, P., Jhanjhi, N. Z., & Luhach, A. K. (2022). Enhanced method of ANN based model for detection of DDoS attacks on multimedia internet of things. Multimedia Tools and Applications, 81(19), 26739-26757.

[15] Karunamurthy, A., Vijayan, K., Kshirsagar, P. R., & Tan, K. T. (2025). An optimal federated learning-based intrusion detection for IoT environment. Scientific Reports, 15(1), 8696.

[16] Ghaleb, S. A., Mohamad, M., Ghanem, W. A. H. M., Alhadi, A. C., Nasser, A. B., & Aldowah, H. (2024). Improved moth search algorithm with mutation operator for numerical optimization problems. Indonesian Journal of Electrical Engineering and Computer Science (IJEECS), 35(2), 8696.

[17] Liu, S., Fauzi, F., & Kok, V. J. (2025). A Systematic Literature Review: Classifying IoT Botnet Data Features Based on Its Lifecycle. IEEE Access.

[18] Bobade, S. Y., Apare, R. S., Borhade, R. H., & Mahalle, P. N. (2025). Intelligent detection framework for IoT-botnet detection: DBN-RNN with improved feature set. Journal of Information Security and Applications, 89, 103961.

[19] Rehman, A., Alharbi, O., Qasaymeh, Y., & Aljaedi, A. (2025). DC-NFC: A Custom Deep Learning Framework for Security and Privacy in NFC-Enabled IoT. Sensors, 25(5), 1381.

[20] Kalidindi, A., & Arrama, M. B. (2025). Feature selection and hybrid CNNF deep stacked autoencoder for botnet attack detection in IoT. Computers and Electrical Engineering, 122, 109984.

[21] Wang, G. G. (2018). Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems. Memetic Computing, 10(2), 151-164.

[22] Ghaleb, S. A., Mohamad, M., Fadzli, S. A., & Ghanem, W. A. H. (2021). Training neural networks by enhance grasshopper optimization algorithm for spam detection system. IEEE Access, 9, 116768-116813.

[23] Ghanem, W. A., & Jantan, A. (2020). Training a neural network for cyberattack classification applications using hybridization of an artificial bee colony and monarch butterfly optimization. Neural Processing Letters, 51(1), 905-946.

[24] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Future Generation Computer Systems, 100, 779-796.

[25] S. UNSW, "The Bot-IoT Dataset," 2021. [Online]. Available: https://research.unsw.edu.au/projects/bot-iot-dataset